

BOA Bilgi Teknolojileri ve Güvenliği olarak sunduğumuz eğitim hizmetleri arasında Zararlı Kod Analizi artan zararlı yazılım tehditleri sebebiyle gün geçtikçe önem kazanmaktadır. Finans ve kamu kurumlarını hedef alan zararlı yazılımlar gün geçtikçe artmakta ve artık kurumların kendilerine özel çözümler geliştirip hem reaktif hem de proaktif davranmaları gerekmektedir.

Sizlere sunduğumuz bu döküman, beş gün süren "Zararlı Kod Analizi" eğitimimizin içerisinden seçtiğimiz bazı kısımlardan oluşmaktadır.

Terminoloji: Malware Kategorileri

Zararlı yazılımlar belli bir amaca ve hedefe yönelik özel olarak hazırlanmaktadır. Her biri farklı amaçlar ile hazırlanan zararlı yazılımlar kendi türlerini oluşturur. Farklı türlerde ki zararlı yazılımlar farklı davranışlar sergiler ve farklı teknikler kullanır. Bazı zararlı yazılımlar tamamen teknik vektörlerden ibarettir, bazıları ise insan unsurunu hedef alarak amacına ulaşır.

Phising / Kimlik Hırsızlığı

Phising yani Kimlik Hırsızlızlığı yada yemleme olarak bilinen yöntem büyük ölçüde insan unsuruna dayalıdır. Saldırgan bu yöntem ile, İnsanları aldatmayı hedefler ve bu sayede kullanıcıların kimlik bilgileri, banka bilgileri, şifreler gibi önemli verilerine ulaşır. Genel olarak "e-posta" yolu ile hedefe ulaşır ve kendisini resmi bir kurumdan geliyormuş gibi gösteren bir elektronik posta hesabı kullanır.

Bu basit yöntem dünya genelinde kötü amaçlı kişiler tarafından oldukça yaygın olarak kullanılmıştır ve sonucunda "milyon dolar" ile ifade edilen ciddi mali kayıplara neden olmuştur.

Riskware

Riskware yazılımları organize olarak hazırlanmış zararlı yazılım türleri arasında yer almaz. Fakat bir dolandırıcılık türüdür, bilgisayara bilinçli veya biliçsiz olarak yüklenir ve sürekli olarak kendini gösterir. Amacı kullanıcıyı bu programı satın almaya zorlamaktadır.

Hoax

Hoax tipi zararlı yazılımlar en popüler zararlı yazılım türleri arasında yer alır. Programcılar tarafından özenle hazırlanan yazılımlardır. Nedeni ise yasal bir yazılımın taklit edilmesi aşamasıdır. Hoax tipi yazılımlar kendilerini anti-virüs, güncelleme (patch) veya belli bir zararlının bilgisayardan silinmesine yönelik bir programmış gibi göstermektedir.

Spyware

Spyware tipi zararlı yazılımlar adından anlaşılacağı gibi casus yazılımlardır. Bu tip zararlı yazılımların amacı kullanıcı/kullanıcılar ve bulaştıkları sistemler hakkında bilgi toplamaktır. Bu bilgiler Kimlik bilgileri, banka bilgileri gibi kişisel bilgiler, internet surfing bilgileri ve sistem bilgileri gibi bilgileri içerir.

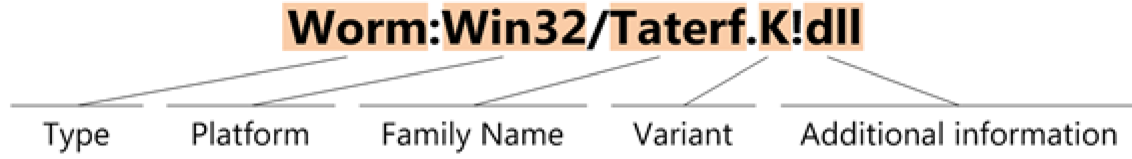
Malformed

Malformed kavramı biraz daha organize saldırılar olarak görülebilir. Genel olarak belli bir dosya tipi içerisine (embed) gömülü olarak gelir. Çalıştırılabilir bir dosya tipi olmadıkları için farkedilmesi zordur. Örnek olarak, PDF dosyalar içerisine gömülü bir zararlı kod bulaştığı sisteme uzaktan erişim sağlayabilir. Bu tip bir zararlının hazırlanması sürecinde sadece programlama bilgisi yetersiz kalır çünkü bu tip bir zararlı dosyanın oluşturulması için mevcut "PDF Parser" içerisinde bir güvenlik açığının bulunması gerekir. PDF uzantılı dosyalar tamamen örnek olarak verilmiştir malformed kod içeren dosya türleri çeşitlilik gösterir. En bilinenleri, *.docx, *.ppt, *.mp3 gibi dosya türleridir.

Zararlı Kod İsimlendirme Standardı

Zararlı kod isimlendirme, anti-virüs firmaları için önemli bir unsurdur. Tespit edilen zararlı yazılımların kategorize edilmesi, türlerine göre ayrılması, platformlarının belirlenmesi gibi süreçlerin bir standardı vardır.

CARO (Computer Antivirus Research Organization) tarafından belirlenen bu isim standardı 1991 yılında yayınlanmış ve 2002 yılında tekrar düzenlenerek anti-virüs firmalarının kullanımına sunulmuştur.



Şekil 1 - CARO isimlendirme standardı.

Resimde görüldüğü gibi birinci kısım zararlı yazılımın türünü, ikinci kısım ise hangi platformu hedeflediğini göstermektedir. Sonraki kısım ise hangi zararlı yazılımın familyasından geldiğini, varyant serisini ve ek bilgileri içermektedir.

Platformlar

İleride temel olarak bahsedileceği gibi zarar yazılım isimlendirme standartlarında platform önemli bir rol oynar. Her zararlı yazılım farklı platformları hedefler ve kategorize işlemide buna göre yapılır. Aşağıda listelenen işletim sistemleri ve platformlar zararlı yazılım isimlendirmede bir etkendir.

İşletim Sistemleri	
AndroidOS	Android işletim sistemi
DOS	MS-DOS platform
EPOC	Psion terminalleri
FreeBSD	FreeBSD platformu
iPhoneOS	iPhone/iOS işletim sistemi
Linux	Linux platform
MacOS	MAC 9.x platformu
MacOS_X	MacOS X veya sonrası
OS2	OS2 platformu
Palm	Palm işletim sistemi
Solaris	System V-tabanlı Unix platformu
SunOS	Unix platforms 4.1.3 veya daha düşük
SymbOS	Symbian işletim sistemi (mobil)
Unix	Genel “unix” platformu
Win16	Win16 (3.1) platformu
Win2K	Windows 2000 platformu
Win32	Windows 32-bit platformu
Win64	Windows 64-bit platformu
Win95	Windows 95, 98 ve ME platformu
Win98	Windows 98 (sadece)
WinCE	Windows CE platformu
WinNT	Windows NT platformu

Programlama ve Script Dilleri

Zararlı yazılımların hangi dilde yazıldıkları analiz sürecini etkileyen bir etken olduğu gibi isimlendirme sürecinde de etkilidir. Aşağıda listelenen programlama ve script dilleri zararlı yazılımları kategorize etmek için önemli bir ölçüttür.

Programlama Dilleri

ABAP	Advanced Business Application Programming scripts
ALisp	ALisp scripts
AmiPro	AmiPro script
ANSI	American National Standards Institute scripts
AppleScript	compiled Apple scripts
ASM	Assembly scripts
ASP	Active Server Pages scripts
AutoIt	AutoIT scripts
BAS	Basic scripts
BAT	BAT scripts
CorelScript	Corelscript scripts
HTA	HTML Application scripts
HTML	HyperText Markup Language scripts
INF	Install scripts
IRC	mIRC/pIRC scripts
Java	Java binaries (classes)
JS	Javascript scripts
LOGO	LOGO scripts
MPB	MapBasic scripts
MSH	Monad shell scripts
MSIL	.Net intermediate language scripts
Perl	Perl scripts
PHP	Hypertext Preprocessor scripts
Python	Python scripts
SAP	SAP platform scripts
SH	Shell scripts
VBA	Visual Basic for Applications scripts
VBS	Visual Basic scripts
WinBAT	Winbatch scripts
WinHlp	Windows Help scripts
WinREG	Windows registry scripts

Makrolar

Makrolar	
A97M	Access 97, 2000, XP, 2003, 2007, and 2010 macros
HE	macro scripting
O97M	Office 97, 2000, XP, 2003, 2007, and 2010 macros - those that affect Word, Excel, and Powerpoint
OpenOM	OpenOffice macros
P98M	Project 98, 2000, XP, 2003, 2007, and 2010 macros
PP97M	PowerPoint 97, 2000, XP, 2003, 2007, and 2010 macros
V5M	Visio5 macros
W1M	Word1Macro
W2M	Word2Macro
W97M	Word 97, 2000, XP, 2003, 2007, and 2010 macros
WM	Word 95 macros
X97M	Excel 97, 2000, XP, 2003, 2007, and 2010 macros
XF	Excel formulas
XM	Excel 95 macros

Dosya Tipleri

Dosya Türleri	
ActiveX	ActiveX controls
ASX	XML metafile of Windows Media .asf files
DOS32	Advanced DOS Extender files
HC	HyperCard Apple scripts
MIME	MIME packets
Netware	Novell Netware files
QT	Quicktime files
SB	StarBasic (Staroffice XML) files
SWF	Shockwave Flash files
TSQL	MS SQL server files
VMSS	Virtual machine suspended state files
XML	XML files

Dosya Uzantıları

Zararlı yazılımlar dosya uzantılarına göre kategorize edilebilirler. Aşağıda verilen dosya uzantıları zararlı kodlarla doğrudan yada dolaylı olarak ilişkili olan dosya türleridir.

Dosya Uzantıları	
.dam	damaged malware
.dll	Dynamic Link Library component of a malware
.dr	dropper component of a malware
.gen	malware that is detected using a generic signature
.kit	virus constructor
.ldr	loader component of a malware
.pak	compressed malware
.plugin	plug-in component
.remnants	remnants of a virus
.worm	worm component of that malware
!rootkit	rootkit component of that malware
@m	worm mailers
@mm	mass mailer worm

Terminoloji: Malware Türleri

Virüs

Virüsler zararlı yazılımlar arasında en çok bilinen türlerdir. Geliştirilme süreci konuya hakim uzman kişiler tarafından işletilir. Klasik programlama dilleri ile değil işlemcinin sunduğu programlama ortamları kullanılarak geliştirilir. Bunun anlamı geliştirilen virüsler platform ve donanım (işlemci) bağımlıdırlar. İşlemcinin sunduğu imkanlar ile programlanmasında ki (low-level) amaç daha optimize kod geliştirme imkanı ve daha küçük boyutta (executable) dosyalar elde etmektir.

Çalıştıkları sistemlerde virüsler çeşitli dosyalara bir enfektor sayesinde bulaşırlar. Virüsler, anti-virüsler tarafından yakalanmamak için çeşitli teknikler kullanırlar. Aynı zamanda virüsler, analist tarafından yapılacak analiz işlemlerini zorlaştırmak için "obfuscation" denen yöntemleri kullanırlar.

Worm (Solucanlar)

Genel olarak virüsler ile aynı teknikleri kullanırlar fakat solucanlar bulaştıkları sistemlerin dahil oldukları ağlara dahil diğer sistemlere bulaşabilmeyi hedefler. Bunun için bünyelerinde ağ servisleri üzerine bulunan zafiyetleri barındırırlar ve mevcut zafiyetler üzerinden diğer sistemlere sızmaya çalışırlar. Düşük seviyeli diller(Assembly) ile değil daha çok C/C++ gibi orta-seviye diller ile geliştirilirler.

Trojan (Truva Atı)

Trojan yada truva atı olarak bilinen bu zararlı yazılımlar sunucu-istemci mantığı ile çalışırlar. Bulaştıkları sistemleri tam anlamıyla saldırgana açarlar ve sistem tamamen saldırganın kontrolüne geçer. Bu süreç zaman zaman komut tabanlı olmakla birlikte saldırgan RDP üzerinden çalışıyormuş gibi sisteme erişim sağlayabilir.

Backdoor (Arka Kapı)

Arka Kapılar saldırganlar tarafında oldukça yaygın olarak kullanılan araçlardır. Diğerlerinden farklı olarak arka kapıların burada ki kullanım amacı farklıdır. Saldırganlar güvenlik açıkları sayesinde sızdıkları sistemlerde kalıcı olabilmek adına arka kapılardan faydalanırlar. Güvenlik açıkları sayesinde zaten sızmış oldukları sistemlere daha sonra tekrar erişebilmek için arka kapıları kullanırlar.

Exploit

Exploit kavramı tüm bahsedilen türlerden farklı bir kavramdır, fakat bir çok firewall ve anti-virus tarafından zararlı yazılım olarak kabul edilirler. Saldırganın belli bir güvenlik açığını kullanarak sisteme erişmesini sağlayan araçlardır. Virüslerden farklı olarak bir çok programlama dili ve scripting dili ile kodlanırlar. Geliştirilmesinde kullanılan teknikler ve yöntemler ileri seviye bilgi gerektirir.

Shellcode (Kabul Kod)

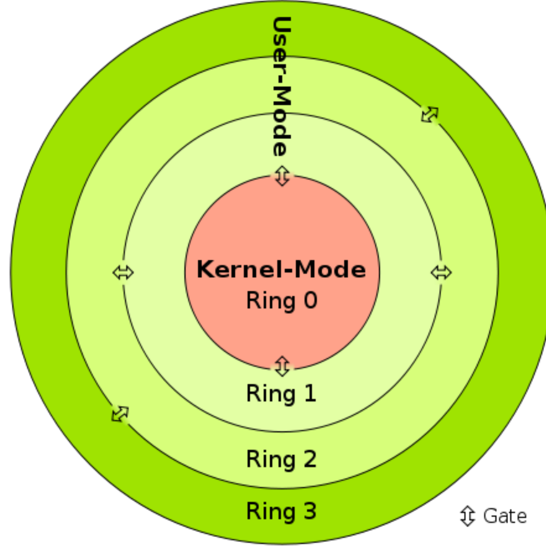
Exploit içerisinde kullanılan "kabuk kodlar" sistemde ki güvenlik açığının kullanılmasından sona çalıştırılacak kodu ifade ederler. Exploit içerisinde gömülü olan ve uzaktan erişimi sağlayan bu kodlar assembly seviyesinde hazırlanırlar ve genelde boyutları 500 byte fazla olmamaktadır.

Rootkit

En tehlikeli zararlı yazılım türüdür, genelde çekirdek seviyesinde çalışmalarından dolayı tespit, analiz ve silme süreçleri oldukça zordur. Diğerlerinden farklı olarak programlama bilgisinin

yanında ileri seviye işletim sistemi ve donanım bilgisi gerektirir. Saldırganın geliştirdiği yani hedeflediği platform hakkında derinlemesine bilgi sahibi olması gereklidir.

Genel olarak 5 tip farklı rootkit türü vardır, herbiri farklı platformları hedefler ve her biri farklı uzmanlık gerektiren konulardır. Stuxnet ve varyantı olan DuQu örneklerinde görüldüğü gibi oldukça karmaşık yapıdadırlar. Bu zararlılar bünyelerinde bir çok güvenlik açığı ve geliştirilmesi esnasında herkesçe bilinmeyen programlama dilleri kullanılmıştır. Bu unsurlar, rootkit adı verilen yazılımların ne derece komplike ve ileri seviye yazılımlar olduğunun göstergesidir.



Şekil 2 - Kernel Mode Katmanı

Kernel Mode (Ring 0)

Bu mod için hazırlanan zararlı yazılımlar çekirdek seviyesinde işletilir ve çekirdek seviyesinde sağlanan fonksiyonların kancalanması ve kullanılması ile sistemin normal işleyişine müdehale eder. Bu modda zararlı kod tespit etmek ileri seviye işletim sistemi çekirdek bilgisi gerektirir. Rootkit yazılımları oluşturduklar trafiği ve sistem üzerinde ki aktivitelerini en üst düzeyde gizlerler.

Geçtiğimiz senelerde ortaya çıkan Stuxnet, Duqu gibi zararlı yazılımlar bu mod çerçevesinde var olan zararlı yazılımların en bilinenleridir.

User Mode (Ring 3)

İşletim sistemi tarafından kullanıcılara sunulan sistem alanı(adres) ve fonksiyonlara erişim sağlarlar. Bu modda çalışan fonksiyonların kancalanması ve amaç doğrultusunda değiştirilmesi ile sisteme müdehale eder.

Hypervisor (VMM)

İşletim sistemleri için geliştirilen sanallaştırma yazılımlarını hedef alırlar. Genel olarak "host" üzerinde çalışan ve donanım kaynaklarını yöneten Hyper-V, VmWare ESX Server gibi sistemleri hedef alırlar. Analiz ve tespit süreci klasik tekniklerden farklı ve daha ileri seviye teknikleri kapsar.

Hardware/Firmware

Donanımsal rootkit kavramı etkili olarak 2008 yılından itibaren daha ileri seviyede konuşulan bir kavramdır. Sistemlere takılan network cihazları, harddiskler ve BIOS gibi donanımlar sayesinde aktif haler gelir. Donanımsal zararlı yazılımların tespiti tüm yöntemlerden farkı olarak işletilmekte ve farklı uzmanlık alanları gerektirmektedir. Firmware bazlı arka kapılar ise genel olarak Modem, Router, Mobil, cihazlar, ATM gibi çok geniş donanımları etkilemektedir.

Saldırı Vektörleri (Attack Vectors)

Zararlı yazılımlar türlerine göre, saldırı vektörleri noktasında farklılık gösterir. Virüslerin türlerinin belirlenmesi, vereceği zararların kestirilmesi, analiz tekniklerinin belirlenmesi sırasında bu vektörlerin anlaşılması önemlidir.

Boot Sector / MBR

Boot sektör genel olarak harddisk üzerinde bulunan bölümler aktif olmadan önce işletilecek komutların yer aldığı bölümdür. Virüsler sistem başlatılırken hangi bölümün aktif edileceği bilgisi yerine kendi işletecekleri komutları bu bölüme işlerler. En temel enfeksiyon belirtisi olarak boot işleminin yavaşlaması ve anlamsız çalıştırılabilir dosyaların boot sonrası varlığıdır.

File Infectors

Zararlı yazılımlar genel olarak bulaştıkları sistemler üzerinde varlıklarını kendilerinden bağımsız kılmak adına diğer dosyalara bulaşırlar. Zararlı yazılımlar içerisinde gömülü olarak bulundurdukları kodları diğer yazılımlara enjekte ederek kendilerini çoğaltırlar. "File Infector" kavramı bir zararlı yazılımın en temel bileşenidir.

Elektronik Posta

Kurumlar için en temel risk sebebi olan ve elektronik postalar sayesinde bulaşan zararlı kodlardır. Temel olarak farklı formatlarda elektronik postalar yoluyla yayılan virüslerdir, aynı zamanda kurumların bünyesinde kullandıkları e-posta servisi üzerinde var olabilecek zafiyetler sayesinde de yayılırlar.

Dosya Paylaşımı

Dosya paylaşımı ile virüslerin yayılması en bilindik yöntemlerden biridir. Video, Müzik gibi çeşitli media ortamları içerisine gömülen zararlı kodların aktif hale gelmesiyle sistemler üzerinde çeşitli işlemlerin yapılabilmesine yol açar.

Bluetooth

Mobil cihazlar üzerinde aktif olan zararlı yazılımların genel olarak gözlemlenen özelliklerinden biridir. Kendiliğinden aktif olan ve çevrede ki diğer mobil cihazlara bulaşmayı hedeflerler. Çeşitli mobil platformların geliştiriciye sundukları "uygulama marketler" çoğalması ile birlikte şu an saldırganlar tarafından çokça tercih edilen bir bulaşma yöntemi değildir.

Web Application (Web Uygulamalar)

Web uygulamalar üzerinden doğrudan veya olası bir güvenlik zafiyeti sayesinde dolaylı olarak bulaşmak için kullanılan yöntemlerden biridir. Bahsi geçen web uygulamalar saldırgan tarafından özel olarak hazırlanan bir uygulama (kit) olmasının yanı sıra tamamen bir probleme dayalı bir bulaşma yöntemi de olabilir.

Enfeksiyon Tespiti ve Zararlı Yazılımın Örneklenmesi

Bir zararlı yazılımın varlığını tespit edebilmek için yukarıda bahsedilen belirtilerden yola çıkmamız gerekir. Aynı şekilde bir zararlı yazılımı analiz edebilmek için belirtilerin sonucunda bir objenin elde edilebilmesi gerekir.

Bu bölümde yukarıdaki belirtilerden yola çıkarak bir zararlı yazılımın nasıl bulunacağı ve sistemin nasıl analiz edileceği anlatılacaktır. Sistem dosyaları, servisler, ağ trafiğ, registry kayıtları gibi zararlının etki edebileceği bir çok noktanın analiz süreci ve araçlar anlatılacaktır.

Kısaca eğer sistemde bir kararsızlık varsa ve buna neden olan şey bir zararlı yazılımsa detaylı bir analiz için hangi noktalara bakmamız gerektiği anlatılacaktır.

Windows Registry

Windows Registry, işletim sisteminin veya kurulu uygulamaların konfigürasyon bilgilerini ve seçenekleri saklayan bölümdür. Eski Windows sürümleri üzerinde bu veriler ".ini" uzantılı dosya içerisinde var oluyordu. Günümüzde ise bu bilgiler yukarıda bahsedildiği gibi "registry" kayıtları üzerinde var olur.

Zararlı kodlar (malwares) registry kayıtlarını kalıcı olabilmek ve kendi konfigürasyonu için kullanır. Zararlı kodlar sistemin başlangıcı yani "boot" esnasında otomatik olarak başlayabilmek için kendini kayıt defterlerine eklerler.

Microsoft dökümantasyonuna göre "Registry" kayıtları, Root Key, subkey, key, value entry ve value/data kısımlarından oluşur. Root Key, bir "registry" kkayıdı için en üst anahtardır. HKEY ile başlayan ve registry kayıtlarında görülen bu kısım "root key" demektir. Subkey ise "root key" altında listelenen bir "alt klasör" niteliğindedir. Key kısmı klasörleri veya değerleri barındırır, subkey ile rootkey anahtarları birlikte anılır.

Registry Rootkeys

Windows üzerinde registry kayıtları arasında beş adet rootkey bulunur. Bunların listesi ve açıklaması aşağıdaki gibidir.

HKEY_LOCAL_MACHINE (HLKM)

İşletim sistemi üzerinde kurulu olan yazılımların ve genel konfigurasyon ayarlarını tutan "root key"dir.

HKEY_CURRENT_USER (HKCU)

Registry üzerinde sadece giriş yapmış ("login") kullanıcıya ait konfigürasyon ayarlarını tutan "root key"dir.

HKEY_CLASS_ROOT

Sistem üzerinde tanımlı verilerin tutulduğu "root key"dir.

HKEY_CURRENT_CONFIG

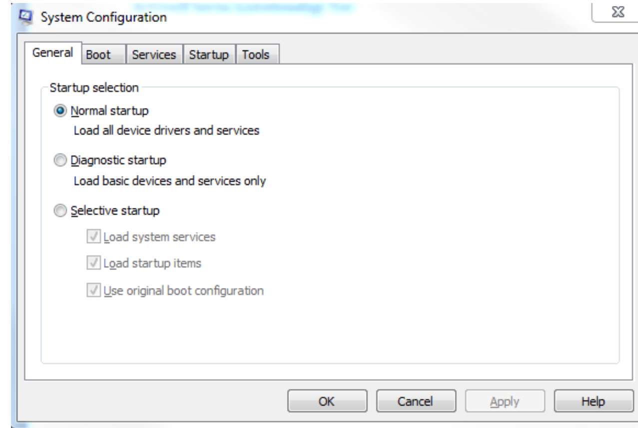
Mevcut donanım konfigürasyonları ve spesifik olan güncel/standart konfigürasyon bilgilerini tutan "root key"dir.

HKEY_USERS

Öntanımlı kullanıcı, yeni kullanıcı ve mevcut kullanıcı için tanımlı ayarların tutulduğu bölümdür.

MSCONFIG

Sistem açılışı esnasında hangi yazılımların otomatik olarak başlayacağı bilgilerini gösteren programdır. Aynı zamanda başlatılan servisler, üretici bilgileri ve servislerin durumu hakkında bilgi edinebilmemizi sağlar.

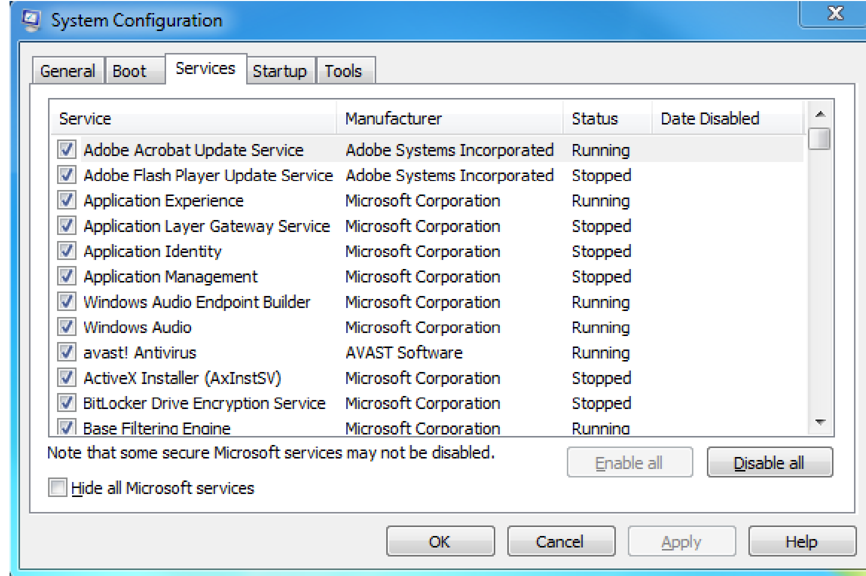


Şekil 3 - Sistem Konfigürasyon Penceresi (MSCONFIG)

Başlatılan servislerin ve ürecitilerin görülmeside "services" sekmesi üzerinden sağlanabilir. Burada farklı ve olağan dışı bir servisin olması "malware" belirtisi olabilir.

Services (Servisler)

Resimde görülen servislerin tamamı bilgisayarımızda kurulu olan yazılımlara ait servisler. Bu servislerin arasında farklı bir servis olması durumunda ilk göze çarpanlar arasında "manufacturer" kısmı olacaktır.



Şekil 4 - MSCONFIG Servisler

Örnek olarak, Çin menşeili zararlı yazılımların kendisini servisler arasında eklemesi sonucunda "Manufacturer" kısmında genelde anlamsız gibi görünen "Çince" karakterler görünmektedir.

Network Trafigi

Network trafiği üzerinde olası problemlerin tespit edilmesi islemi için çeşitli analiz araçları vardır. Bunlardan bazıları Wireshark, TcpView gibi yazılımlardır, otomatize edilmiş bir şekilde analiz etmenin dışında bazı sistem dosyalarının "manual" olarak incelenmesi gerekebilir.

WireShark

Gelişmiş bir "network analiz" aracı olan wireshark platform olarak Windows, Linux, MacOS ve Solaris gibi işletim sistemlerini destekler. Tüm ağ kartları üzerinde TCP/IP verilerini analiz edebilir. Wireshark desteklediği 750'nin üzerinde protokolü analiz edebilir.

Paketleri yakalayabilir ve analiz için bir dosyaya kaydedebilir aynı şekilde formatı uyduğu takdirde bir dosyayı açabilir. Gerçek zamanlı olarak analiz edebilme yeteneği vardır ve filtre özelliğini bünyesinde barındırır. Örnek olarak, sadece "HTTP" verilerini görebilmek için ilgili protokolü yada belli bir IP adresini filtreleyebilir.

NetStat

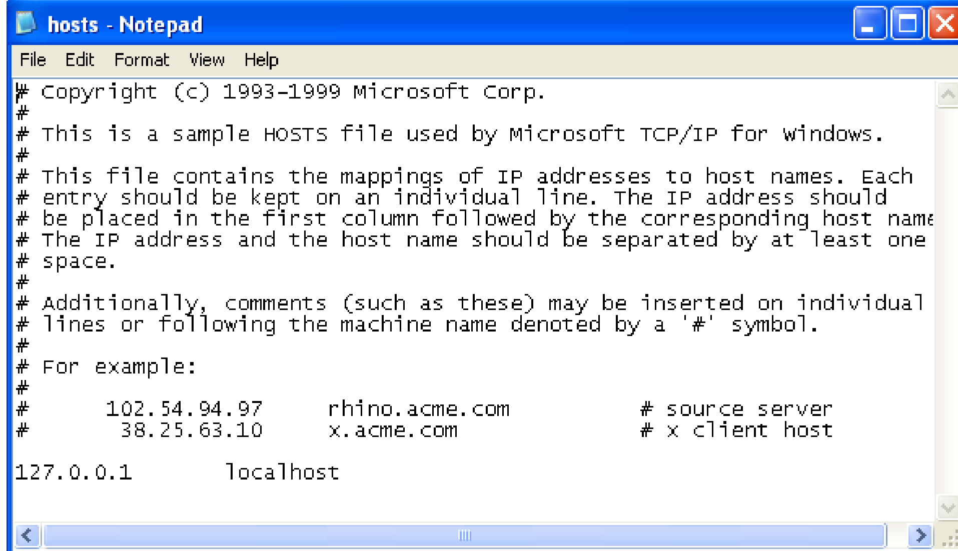
Bir komut satırı aracı olmakla birlikte bir çok platform üzerinde çalışabilir haldedir. Windows işletim sistemi üzerinde "default" olarak gelir. Yönlendirme tablıları ve ağ arayüzü istatistiklerini görüntünmesini sağlayan bir araçtır. Netstat komutu ağdaki problemleri bulma ve ağ üzerindeki trafiğin miktarını belirlemek için kullanılabilir. İlerleyen bölümlerde anlatılacağı gibi "netstat" yazılımı ile zararlı kod tespiti (gelişmiş bir zararlı yazılım değilse) mümkündür.

TcpView

Sysinternals suite içerisinde yer alan bir yazılımdır ve bu yazılım sayesinde anlık olarak aktif bağlantıları görebilmek mümkündür. Herhangi bir bağlantıyı buradan takip edilebilir ve sonlandırılabilir.

Hosts Dosyası

Linux, Windows gibi işletim sistemleri üzerinde “hostname” verilerini “IP” adrese göre düzenleyen dosyadır. İçeriği tamamen düz metindir ve burada yapılacak değişiklikler ile mevcut sistemi farklı adreslere yönlendirilebilir.




```
# Copyright (c) 1993-1999 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

127.0.0.1       localhost
```

Şekil 5 - Windows "hosts" Dosyası

Zararlı yazılımlar bu dosya üzerinde yapacakları değişiklikler ile mevcut alan isimlerini kendi belirlediği adreslere yönlendirebilirler. Bu sayede bulaştıkları sistem üzerindeki kullanıcılar çeşitli zararlı sitelere yönlendirilebilirler.

Dosya Sistemi

Dosya sistemi üzerinde var olan değişikliklerin tespit edilmesi önemlidir. İlerleyen bölümlerde bahsedilecek olan Windiff yazılımı sayesinde dosya sistemi üzerinde yapılan değişiklikleri görebilmneniz mümkündür. Belirli dizinler altına eklenen yeni dosyalar ve bu dosyaların ne işe yaradığı gibi verileri elde etmek için analiz öncesi yapılan işlemdir.

Bunların dışında sistem üzerinde kendini gizlemeyi başaran zararlı yazılımlar mevcuttur. Bu sayede analiz sürecinin geciktirilmesi ve dosya farkedilmediği sürece işlevlerin devam etmesi sağlanır. Sistem üzerinde ki zararlı yazılımları tespit edebilmek için gizli dosyaların mutlaka incelenmesi gerekir.

İlerleyen bölümlerde hem dinamik hemde statik olarak analiz edilen zararlı yazılımın bu tip bir özelliği vardır. Bunun sistem üzerinde tespit edilebilmesini sağlamak için komut satırı aracından faydalanmamız gerekir.

C:\WINDOWS\system32\cmd.exe

```
C:\>dir /s /a:h
 Volume in drive C has no label.
 Volume Serial Number is B8DC-A642

 Directory of C:\

05.11.2012  06:07               142 autorun.inf
07.11.2012  00:11               252 boot.ini
30.10.2012  10:23                 0 IO.SYS
30.10.2012  10:23                 0 MSDOS.SYS
12.08.2004  06:02            47.564 NTDETECT.COM
12.08.2004  06:02           250.032 ntldr
13.11.2012  10:28     1.610.612.736 pagefile.sys
30.10.2012  00:47    <DIR>          RECYCLER
20.10.2012  19:37            26.112 Scan with Antivirus.exe
30.10.2012  10:28    <DIR>          System Volume Information
               8 File(s)  1.610.936.838 bytes

 Directory of C:\DELL

11.11.2012  23:55             6.656 Thumbs.db
               1 File(s)          6.656 bytes

 Directory of C:\Documents and Settings
```

Şekil 6 - Gizli dosyaların tespit edilmesi.

Resimde görüldüğü üzre “c:” sürücüsü içerisindeki gizli dosyalar listelenmiştir. Bunu yapabilmek için “dir” komutuna verilen “/s /a:h” parametreleri kullanılmış ve “Scan with Antivirüs” isminde bir şüpheli dosya tespit edilmiştir.

Kullanıcı Hesapları

Zararlı yazılımların sisteme uzaktan erişimi mümkün kılabilecek çeşitli teknikleri vardır. Bunlardan biri sistem üzerinde yeni bir kullanıcı veya kullanıcı grubu oluşturmaktır. Bu tip bir etkinlik bir zararlı yazılım belirtisi olabilir ve bundan dolayı kontrol edilmesi gereken noktalardan biridir.

C:\WINDOWS\system32\cmd.exe

```
C:\>net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the compu
ter

Members

-------------------------------------------------------------------------------
Administrator
infected
The command completed successfully.


C:\>net users

User accounts for \\INFECTED-C62F8A

-------------------------------------------------------------------------------
Administrator            Guest                    HelpAssistant
infected                 SUPPORT_388945a0
The command completed successfully.

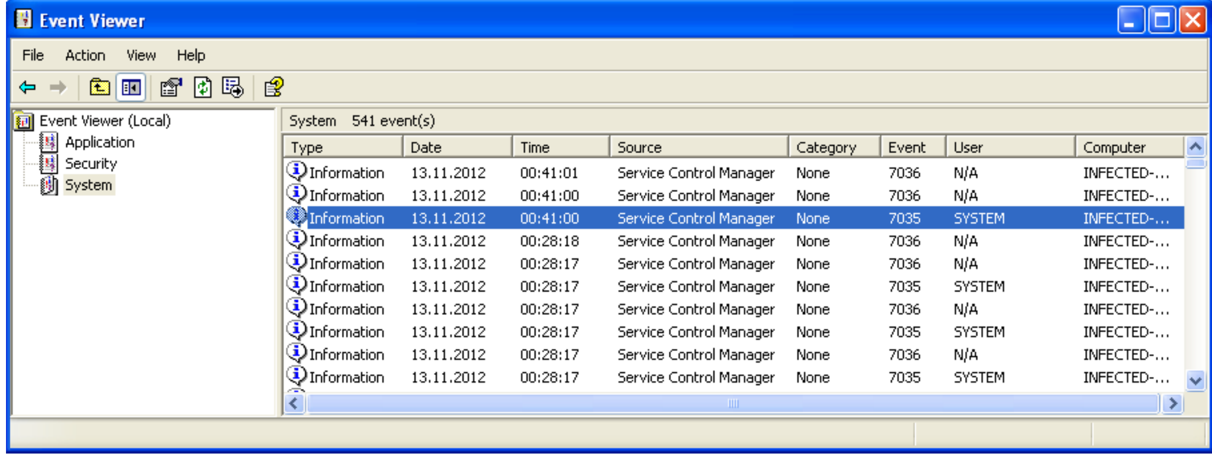

C:\>_
```

Şekil 7 - Sistem üzerinde ki kullanıcı hesapları ve gruplar.

Yukarıda görüldüğü gibi analiz edeceğimiz zararlı yazılım sistem üzerinde herhangi bir kullanıcı oluşturmamıştır. Administrator grubunun içeriği olması gerektiği gibidir, bunun yanında sistem üzerinde ki kullanıcılar üzerinde de herhangi bir değişiklik gözlemlenmemektedir.

Olay Kayıtları (Event Log)

Windows tarafından sunulan yazılımlardan biridir. Sistem üzerinde yapılan değişiklikler 3 kategoride (loglanır) kaydedilir. Bunlar sırasıyla uygulama günlüğü, güvenlik günlüğü ve sistem günlüğü olarak incelenir.



Şekil 8 - Event Log (Olay Günlüğü)

Windows olay kayıtları içerisinde dikkat edilmesi gereken girdiler şu şekildedir. Örnek olarak dikkat edilmesi gerekenler kullanıcı girişleri ve çıkışları (logon/logoff), kullanıcı hesaplarındaki değişiklikler, şifre değişiklikleri, servis başlatıldı yada durduruldu mesajları, "object access denied" gibi mesajlardır.

Reverse Engineering (Tersine Mühendislik)

Reverse Engineering yani Tersine Mühendislik kavramı hemen her meslek ile ilgili bir kavramdır. Herhangi bir şeyin "nasıl çalıştığı" ve "nasıl yapıldığı" bilgisini elde etmek için yapılan sürece verilen isimdir.

Bir ürünün veya yazılımın nasıl çalıştığı, yapısının anlaşılması için detaylı olarak her bir bileşeninin incelenmesi/sökülmesi sürecini kapsar. Tersine Mühendislik hangi amaçlar ve hangi platformlar için yapılırsa yapılsın tekniklerin ve ürünün iyi tanınması gerekir.

Tersine Mühendisliğin ne denli önemli olduğunu anlamak için "Çin Halk Cumhuriyetin"nin son yıllarda ki ekonomik büyümesini neye borçlu olduğunu anlamak yeterlidir. Yazılım, Elektronik ve otomativ sanayii gibi bir çok alanda yapılan ve ekonomiğe olumlu anlamda katkısı olan hemen her parametre "reverse engineering" sürecine dayanır.

Reverse Code Engineering (RCE)

"Reverse Engineering" konusu çok kapsamlıdır. Yazılımlar üzrerinde gerçekleştirilecek tersine mühendlisk kavramı genel olarak "Reverse Code Engineering kısaca RCE" olarak adlandırılır.

Kapalı kaynak kod bir yazılımın (zararlı yazılımlar, ticari yazılımlar vb.) yapısının anlaşılması "tersine mühendislik" yapmaktan geçer. Reverse Engineer (tersine mühendis) olarak adlandırılan kişilerin bu yola başvurmalarında ki amaç aşağıda ki şekildedir.

Güvenlik Analizi

Kaynak kodlarına sahip olunmayan yazılımların içerisinde var olabilecek güvenlik açıklarını incelemek için kullanılır. Kaynak kodlarına sahip olsak dahi her bir satırın incelenmesinden ise temel zafiyetlerin canlı olarak takibinin yapılabilmesi daha kolay bir işlemdir. Bu nedenle yazılımlar "reverse" edilerek olası güvenlik problemleri tespit edilir.

Dökümantasyonu Olmayan Bileşenler (Undocumented Functions)

Yazılımların diğer programcılar tarafından anlaşılabilmesi için en önemli unsur dökümantasyonunun iyi yapılabilmesidir. Uygulamalara ait fonksiyonların ne iş yaptığı ve hangi parametrelere dayalı olarak çalıştığı bilgileri detaylı olarak dökümantasyonlarda yer alır. Örneğin, Windows işletim sisteminin sunduğu API bilgileri MSDN (Microsoft Developers Network) içerisinde yer alır.

Undocumented (dökümantasyonu olmayan) API bilgilerinin anlaşılması ve çalışma mantığının çözülebilmesi için "Reverse Engineering" süreçleri aktif rol oynar. Zararlı kod analizinde programcı tarafından yazılan fonksiyonların çözülebilmesi için "Reverse Engineering" zorunludur.

Yazılım Mimarisinin Anlaşılması

Yazılımların yapısının anlaşılması ve akış diagramının çıkarılabilmesi için işletilmesi gereken süreçlerin tamamı "Reverse Engineering" ile ilgilidir. Yazılımın "neyi nasıl yaptığı" bilgisinin anlaşılabilmesi için yazılımın "reverse" edilebilmesi gereklidir. Akış diagramının çıkartılması, hangi dosyalar ile iletişim halinde olduğunun anlaşılması ve (varsa) donanımlar ile nasıl haberlerleştiğinin çözülebilmesi "iyi derecede tersine mühendislik" bilgisi gerektirir.

Derlenmiş bir kod içerisinde kullanılan veri yapılarının, tanımlanan değişken bilgilerinin görülebilmesi için makine kodu seviyesinde inceleme yapılmalıdır.

Kapalı Kaynak Kod (Closed Source Code)

Ticari yazılımların hemen hepsi kaynak kodları ile birlikte gelmez ve paylaşılması risklidir. Kapalı kaynak kod yazılımların analiz edilebilmesi sadece "tersine mühendislik" ile mümkündür. Derlenebilir hangi dil ile yazılmış olursa olsun yazılımları makine kodu seviyesinde incelemek mümkündür.

Hata Ayıklama (Bug Fixing)

Yazılım içerisinde var olabilecek sorunların belirlenebilmesi için "kaynak kod analiz" işlemi yapmaktansa "live trace" işleminden geçirilmesi vakit tasarrufu anlamına gelir. Programın tüm kaynak kodunun analiz edilmesinden ise çalışma anında oluşan hataları "one-shot" olarak yakalamak daha az zaman gerektirir.

Yama Analizi (Patch Analysis)

Bir çok yazılım üreticisi sundukları yazılımlarda oluşan güvenlik yada genel problemlerini giderebilmek için "patch" yani "yama" olarak hazırlanan küçük eklentilerden yararlanırlar. Programın doğrudan derlenmiş koduna (binary) müdehale ederek genelde "bir kaç instruction" veya "farklı fonksiyonları" değiştirerek mevcut problemleri giderirler.

Yama Analizi, zararlı kod yazan yada güvenlik problemlerinden faydalanarak sistemlere sızmak isteyen kişilerin fazlaca önem verdikleri bir konudur. Örnek senaryo olarak Microsoft firması MS Office üzerinde kendi bünyesinde bulduğu bir güvenlik problemi için bir "patch" yayınlamış olsun.

Bu yayınlanan "yama" sadece sistemlerini sürekli güncel tutan kişileri saldırganlara karşı korur. Açığın nerede olduğunun ve nasıl "exploit" edildiğinin anlaşılabilmesi için "yama analizi" yapılabilmesi gereklidir.

Böylece saldırgan bu açığın hangi problemlerden kaynaklandığını ve nasıl kullanılabileceği bilgisini kolaylıkla elde edebilir. Bunun için ilgili "patch" dosyasının "reverse" edilebilmesi gereklidir.

Statik Analiz – DEMO

```
 686p
 mmx
 model flat


; Segment type: Pure code
; Segment permissions: Read/Write/Execute
_text segment para public 'CODE' use32
assume cs:_text
;org 401000h
assume es:nothing, ss:nothing, ds:_text, fs:nothing, gs:nothing


public start
start proc near
call    $+5
```

Şekil 9 - EXE Segment İzinleri

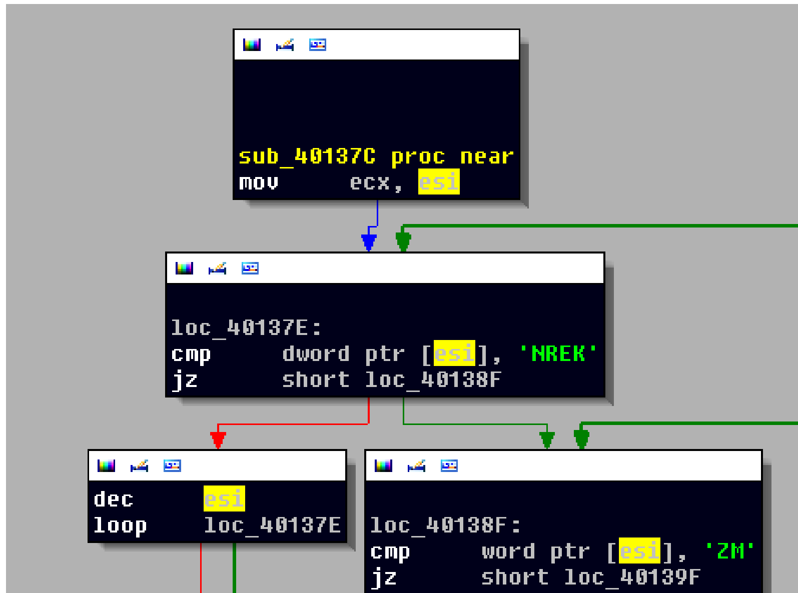
Yukarıda örnek zararlı kodun segment yapısı temel olarak görülüyor. Ekran görüntüsünde ilk bakışta dikkat çeken "segment permissions" kısmıdır. Normal derlenmiş bir yazılımın "segment hakları" müdehale edilmediyse "read/execute" olarak belirlenir. Fakat burada zararlıyı kodlayan programcı "text" segmenti için ayrıca "writeable" hakkını vermiş.

Bunun sebebi text segment yani kod segment üzerinde zararlı kodun yapacağı değişikliklerdir. Normalde bu segment üzerinde değişiklik yapabilmek için özel işlemler gerçekleştirilir. Burada analiz ettiğimiz zararlı kod "antivirüslerin" kafasını karıştırmak için "polimorfizm" denen teknik kullanılmıştır.

```
loc_401005:
pop     ebp
sub     ebp, 401005h
mov     ss:dword_401CBF[ebp], esp
mov     eax, 60C4h
mov     ss:dword_401CC7[ebp], eax
mov     esi, [esp+8]
call    sub_40137C
call    sub_40125A
call    sub_4013AF
call    sub_401039
jmp     loc_401CCB
start endp
```

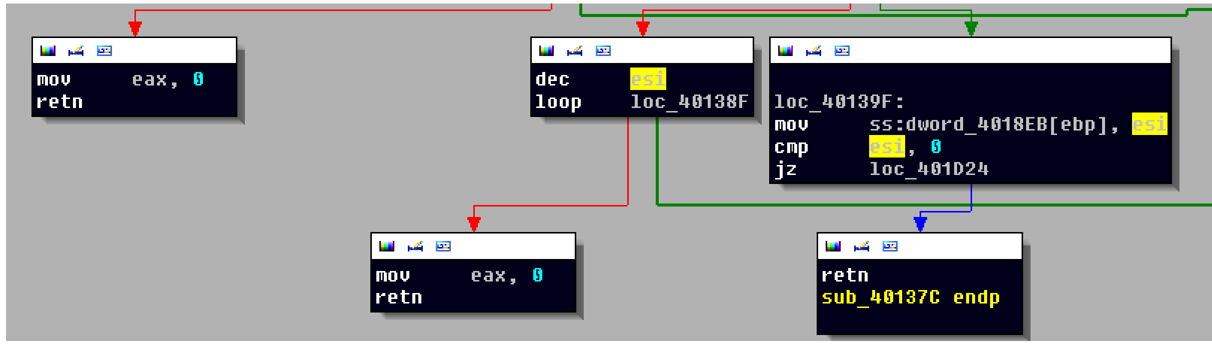
Şekil 10 - main fonksiyonu

Yukarıda görüldüğü gibi zararlının başlama anında “CALL” komutu ile 4 fonksiyon çağrılıyor. Bu 4 fonksiyonun ne işe yaradığı tek tek incelenecektir. Eğer sırasıyla incelemek gerekirse ilgili fonksiyonun üzerine gelip “ENTER” tusuna basıldığında gerekli komutları ve fonksiyona ait diyagramı görebiliriz.



Şekil 11 - API Giriş Noktalarının Hesaplanması

Daha önce “shellcode” kavramından bahsetmiştik, shellcode içerisinde eğer kullanacağınız API var ise bunların “entrypoint” konumlarını hesaplamak zorundasınız. Aynı şey zararlı içerisinde kullanılan API(ler) için de geçerlidir.



Şekil 12 - Kernel32.dll base-address hesabı

Yukarıda gösterilen resim bir öncekinin devamı niteliğindedir. Hafıza alanı içerisinde (in-memory) "KERNEL32.DLL" için arama rutini ve diyagramı görülüyor. Yukarıda görüldüğü gibi "EAX Register" taşıdığı değer "1" ise kernel alanı bulunamiyor bu durumda gerekli rutinler tekrar işletilir, eğer (eax=esi) değeri "0" ise kernel alanı bulunur ve gerekli API'ler için "entry-point" adresleri hesaplanır. Tüm rutinler tamamlanıyorsa kod "sub_401d24" adresine dallaniyor, sözkonusu adresin üzerine gelip "enter" tusuna basip gerekli rutini incelemeye baslayabiliriz.

```
.text:00401D24 loc_401D24:                             ; CODE XREF: sub_40137C+2C↑j
.text:00401D24                                         ; sub_4013AF+24B↑j ...
.text:00401D24                 call    sub_40207A
.text:00401D29                 push    0
.text:00401D2B                 call    ss:dword_401847[ebp]
.text:00401D31                 cmp     eax, 0
.text:00401D34                 jz      loc_401E22
.text:00401D3A                 push    0Fh
.text:00401D3C                 call    ss:dword_40186F[ebp]
.text:00401D42                 cmp     eax, 0
.text:00401D45                 jz      loc_401E22
.text:00401D4B                 mov     ss:dword_4070C0[ebp], eax
.text:00401D51                 push    0
.text:00401D53                 push    0
.text:00401D55                 push    0FFFFFFFFh
.text:00401D57                 push    eax
.text:00401D58                 call    ss:dword_401881[ebp]
.text:00401D5E                 dec     eax
.text:00401D5F                 mov     ss:dword_406EF5[ebp], eax
.text:00401D65                 cmp     eax, 0
.text:00401D68                 jb      loc_401E22
.text:00401D6E
.text:00401D6E loc_401D6E:                             ; CODE XREF: sub_4013AF+A6E↓j
.text:00401D6E                 push    104h
.text:00401D73                 mov     ebx, offset byte_406EF9
.text:00401D78                 add     ebx, ebp
00000F3A 00401D3A: sub_4013AF+98B
```

Şekil 13 - Registry Rutinini Çağrısı

İlgili fonksiyonun içerisine girdiğimizde resimde görüldüğü gibi fonksiyonun ilk komutu bir başka fonksiyonu çağırıyor. Bu fonksiyonun ne olduğunu anlayabilmemiz için ilgili fonksiyonu yeni pencerede ("ALT-ENTER") açıyoruz.

```
.text:00402081 ; ---------------------------------------------------------------------------
.text:00402086 aSoftwareMicros db 'SOFTWARE\Microsoft\Windows\CurrentVersion\Run',0
.text:00402086                                         ; DATA XREF: sub_40207A+1C7↓o
.text:00402086                                         ; sub_40207A+28B↓o
.text:004020B4 aReg_sz         db 'REG_SZ',0           ; DATA XREF: sub_40207A+1BD↓o
.text:004020BB dword_4020BB    dd 0                    ; DATA XREF: sub_40207A+1AF↓o
.text:004020BB                                         ; sub_40207A+212↓r ...
.text:004020BF byte_4020BF     db 0                    ; DATA XREF: sub_40207A+1A7↓o
.text:004020C0                 dd 0Eh dup(0)
.text:004020F8                 db 0
.text:004020F9 dword_4020F9    dd 32h                  ; DATA XREF: sub_40207A+1FC↓o
.text:004020F9                                         ; sub_40207A+268↓w ...
.text:004020FD unk_4020FD      db    0                 ; DATA XREF: sub_40207A+1F0↓o
.text:004020FD                                         ; sub_40207A+229↓o
.text:004020FE                 db    0
.text:004020FF                 db    0
.text:00402100                 dd 40h dup(0)
.text:00402200                 db 0
.text:00402201 dword_402201    dd 104h                 ; DATA XREF: sub_40207A:loc_402262↓o
.text:00402201                                         ; sub_40207A+230↓r ...
.text:00402205 dword_402205    dd 0                    ; DATA XREF: sub_40207A+20C↓r
.text:00402205                                         ; sub_40207A:loc_4022D1↓w ...
.text:00402209 ; ---------------------------------------------------------------------------
.text:00402209
.text:00402209 loc_402209:                             ; CODE XREF: sub_40207A+7↑j
```

Şekil 14 - Registry Verileri

İlk satırdan da anlayabileceğimiz gibi bu bir kayıt defteri (registry) rutini ve bu rutinin işletilmesinin sebebi zararlı kodun kendini kayıt defterine ekleyerek otomatik olarak çalışmasını sağlamak. Amacımız virüsün yapısının statik olarak anlaşılması, bu süreçler ise "davranışsal" olarak tespit edilebilir. Bizi ilgilendiren kısımlar "tamamen statik" olarak yapının anlaşılmasıdır, bu nedenle bu kısım daha sonra incelenecektir.

```
.text:00401D24 loc_401D24:                             ; CODE XREF: sub_40137C+2C↑j
.text:00401D24                                         ; sub_4013AF+24B↑j ...
.text:00401D24                 call    sub_40207A
.text:00401D29                 push    0
.text:00401D2B                 call    ss:dword_401847[ebp]
.text:00401D31                 cmp     eax, 0
.text:00401D34                 jz      loc_401E22
.text:00401D3A                 push    0Fh
.text:00401D3C                 call    ss:dword_40186F[ebp]
.text:00401D42                 cmp     eax, 0
.text:00401D45                 jz      loc_401E22
.text:00401D4B                 mov     ss:dword_4070C0[ebp], eax
.text:00401D51                 push    0
.text:00401D53                 push    0
.text:00401D55                 push    0FFFFFFFFh
.text:00401D57                 push    eax
.text:00401D58                 call    ss:dword_401881[ebp]
.text:00401D5E                 dec     eax
.text:00401D5F                 mov     ss:dword_406EF5[ebp], eax
.text:00401D65                 cmp     eax, 0
.text:00401D68                 jb      loc_401E22
.text:00401D6E
.text:00401D6E loc_401D6E:                             ; CODE XREF: sub_4013AF+A6E↓j
.text:00401D6E                 push    104h
.text:00401D73                 mov     ebx, offset byte_406EF9
.text:00401D78                 add     ebx, ebp
00000F3A 00401D3A: sub_4013AF+98B
```

Şekil 15 - Clipboard İşlemleri

Artık ilk "call" komutu ile hangi fonksiyonun çağrılacağı ve ilgili fonksiyonun ne iş yaptığını öğrendiğimize göre geri kalan satırlar ile ilgilenebiliriz. İlk işaretli alan "OpenClipboard" fonksiyonunu kullanmak için yapılan çağrı ve parametreleri içerir.

NOT: *OpenClipboard ve benzeri bir çok API hakkında gerekli bilgileri ezbere bilmek çoğu zaman mümkün değildir. Bunun için "Microsoft MSDN" sayfalarını inceleyerek hangi API ne tür değerler alıyor ve ne işe yarıyor bilgisini elde edebilirsiniz. (http://msdn.microsoft.com/)*

Bir sonra ki isaret edilen alanda ise "GetClipboardData" çağrılıyor, görevi "clipboard" içerisinde ki verileri almak olarak açıklanabilir. "OpenClipboard" fonksiyonunun çağrılma sebebi daha net olarak anlaşılıyor. Nedeni ise, "GetClipboardData" fonksiyonunu kullanmak için "OpenClipboard" fonksiyonu panoyu (clipboard) daha önce açmış olmak gerekir.

Bir sonra ki işaretli alan ise "DragQueryFile" fonksiyonunu çağırır. Burada aldığı deperlerin stack alanına "push" edildikleri görülüyor. Aldığı son değer "0xFFFF" olduğunda, dönüş değeri olarak geriye "drop" edilen dosyaların sayısı döndürülür.

NOT: *Buraya kadar yapılan işlemler sonucunda zararlı yazılımın "clipboard" üzerinde çeşitli değişiklikler yaptığını anlıyoruz.*

Her API kullanımı sonrasında mantıksal bir kıyaslama yapılmış burada "clipboard" üzerinde işlem yapmak mümkün değilse ne yapılacağı ile ilgili rutinler var. Bu kısmı ilerleyen zamanlarda inceleyeceğiz.

Otomatize Malware Analiz

Zararlı kod analizinin dinamik olarak yapılması işlemini otomatize eden araçlar günümüzde hızla yaygınlaşmaktadır. Antivirüs firmaları "sandbox" adı verdikleri platformları ürün haline getirip büyük firmalara satmakdır.

Dinamik olarak analiz işlemi bir çok kritik veriyi elde etmemize neden olur. Bu verilerin analiz süreci tamamlanması ve zararlının bulaştığı sistemden nasıl kaldırılacağı hakkında analist için yol haritası niteliğindedir.

Örnek olarak, bu "sandbox" araçları zararlı kodun "registry" operasyonları, "load time" ve "run-time" yüklenen "kütüphaneler" hakkında detaylı bilgiler sunabilirler.

- Load-time Dlls

Module Name	Base Address	Size
C:\WINDOWS\system32\ntdll.dll	0x7C900000	0x000AF000
C:\WINDOWS\system32\kernel32.dll	0x7C800000	0x000F6000

- Run-time Dlls

Module Name	Base Address	Size
C:\WINDOWS\system32\NETAPI32.dll	0x5B860000	0x00055000
C:\WINDOWS\system32\comctl32.dll	0x5D090000	0x0009A000
C:\WINDOWS\system32\WS2HELP.dll	0x71AA0000	0x00008000
C:\WINDOWS\system32\WS2_32.dll	0x71AB0000	0x00017000
C:\WINDOWS\system32\wsock32.dll	0x71AD0000	0x00009000

Şekil 16- Anubis DLL İşlemleri

Resimde görülen ekran bir sandbox analiz sonucudur. En çok kullanılan "sandbox" sistemlerinden biri olan Anubis ücretsiz bir servistir. Zararlı kodun "local" sistem üzerinden yada bir "URL" aracılığı ile yüklenerek analiz edilebilmesine olanak tanır.

- Registry Values Read:

HKLM\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers\0\Hashes\{94e3e076-8f53-42a5-8411-085bcc18a68d}	ItemData	0xbd9a2adb42ebd8560e250e4df8162f67	1
HKLM\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers\0\Hashes\{94e3e076-8f53-42a5-8411-085bcc18a68d}	ItemSize	229	1
HKLM\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers\0\Hashes\{94e3e076-8f53-42a5-8411-085bcc18a68d}	SaferFlags	0	1
HKLM\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers\0\Hashes\	HashAlg	32771	1

- Monitored Registry Keys:

Key Name	Watch subtree	Notify Filter	Count
HKLM\system\CurrentControlSet\control\NetworkProvider\HwOrder	0	Value Change	1

Şekil 17 - Anubis Registry İşlemleri

Yukarıda görülen ekranda "zararlı yazılımın" registry kayıtları üzerinde yaptıkları operasyonlar görülmekte. Çalıştıktan sonra düzenlenen registry kayıt bilgileri ile zararlının bazı operasyonları iptal edilebilir.

Linux Payload: Uygulama Shellcode Analiz ("Radare")

Daha öncede bahsettiğimiz gibi yukarıda bahsedilen virüs "destructive" bir payload'a sahip değildir fakat zararlı kodların çoğu "destructive" bir "payload" içerir. Yani genelde bu verinin çalınması yada bulaştığı sisteme zarar veren bir kod parçası olur.

Peki bu kodlar nasıl incelenir ? Linux üzerinde bu zararlıların incelenmesi için bir çok araç ve yöntem mevcuttur. Burada araçların kullanımı öğrenmek adına "radare" isimli yazılımdan bahsedilecektir. Radare komut satırı üzerinden çalışan bir yazılımdır ve disassembler olarak kullanılır.

Elimizde bir "payload/shellcode" var ve bunun ne iş yaptığını bilmiyoruz. Bunu analiz etmek için aşağıda ki adımları takip etmemiz gerekiyor. Analiz aşaması tamamen statik olarak gerçekleşecektir.

```
yasin@yasin-VirtualBox:~/pilot$ radare shell
open ro shell
> Importing file information...
[Information]
class=ELF32
enconding=2's complement, little endian
os=linux
machine=Intel 80386
arch=intel
type=EXEC (Executable file)
stripped=No
static=No
baddr=0x08048000
[Entrypoint]
Memory address: 0x080483a0
> Importing symbols...
30 sections added
12 fields added
5 imports added
30 symbols added
84 strings added
> Analyzing code...
  [-] 24:02:04:00 ====
strings: 84
functions: 15
structs: 0
data_xrefs: 39
code_xrefs: 4
[0x080483A0]>
```

Şekil 18 - Radare Anaekran

Yukarıda görüldüğü gibi “payload/shellcode” yazılıma yüklendikten sonra “Radare2” içerisine yani kendi komut satırına düşüyoruz. Bu programımızın “payload” yüklendikten sonra ki “anaekranıdır.”

```
[0x080482f0]> f
0x08049cd4 0 section_end..strtab
0x08049a88 588 section..strtab
0x08049a88 0 section_end..symtab
0x08049658 1072 section..symtab
0x080491a5 0 section_end..shstrtab
0x0804909f 262 section..shstrtab
0x0804909f 0 section_end..comment
0x08049034 107 section..comment
0x0804a038 0 section_end..bss
0x0804a034 4 section..bss
0x0804a034 0 section_end..data
0x0804a014 32 section..data
0x0804a014 0 section_end..got.plt
0x0804a000 20 section..got.plt
0x0804a000 0 section_end..got
0x08049ffc 4 section..got
0x08049ffc 0 section_end..dynamic
0x08049f14 232 section..dynamic
0x08049f14 0 section_end..jcr
```

Şekil 19 - Section Listesi

Yukarıda görüldüğü gibi “radare” içerisinden “section list” penceresi görülebilir. Burada bizim “disassemble” etmek istediğimiz fonksiyon “main” fonksiyonudur. Doğrulamak adına “pd” komutunu vererek “main” fonksiyonunun hangisi olduğunu görmemiz gerekiyor.

```
[0x080482f0]> pd
      0x080482f0  section..text:
      0x080482f0    31ed            xor ebp, ebp             ; [13]
      0x080482f2    5e              pop esi
      0x080482f3    89e1            mov ecx, esp
      0x080482f5    83e4f0          and esp, 0xfffffff0
      0x080482f8    50              push eax
      0x080482f9    54              push esp
      0x080482fa    52              push edx
      0x080482fb    6870840408      push dword sym.__libc_csu_fini
      0x08048300    6800840408      push dword sym.__libc_csu_init
      0x08048305    51              push ecx
      0x08048306    56              push esi
      0x08048307    68dc830408      push dword sym.main
      0x0804830c    e8cfffffff      call dword imp.__libc_start_main
         ; imp.__libc_start_main()
      0x08048311    f4              hlt
      ; ------------
      0x08048312    6690            o16 nop
      0x08048314    6690            o16 nop
      0x08048316    6690            o16 nop
      0x08048318    6690            o16 nop
      0x0804831a    6690            o16 nop
      0x0804831c    6690            o16 nop
      0x0804831e    6690            o16 nop
      0x08048320  sym.deregister_tm_clones:
      0x08048320    b837a00408      mov eax, 0x804a037
      0x08048325    2d34a00408      sub eax, sym.completed.6382
      0x0804832a    83f806          cmp eax, 0x6
      0x0804832d    7702            ja 0x8048331
      0x0804832f    f3              invalid
```

Şekil 20 – Ful assembly dump verisi

EK - Windows “Crash Dump” Analizi

Windows işletim sisteminde oluşan “mavi ekran” yani “BSOD” (Blue Screen of Death) hataları hemen her kullanıcı için tanıdıktır. Hatanın çözümü çoğu zaman işletim sisteminin yeniden kurulması olarak telafi edilir. Kritik sistemler için aynı durum söz konusu değildir, sistem üzerindeki “driver” problemleri bu sonuca sebep olabileceği gibi işletim sistemi çekirdek seviyesinde çalışan “rootkit” adı verilen zararlı yazılımlar’da bu duruma sebep olabilirler.

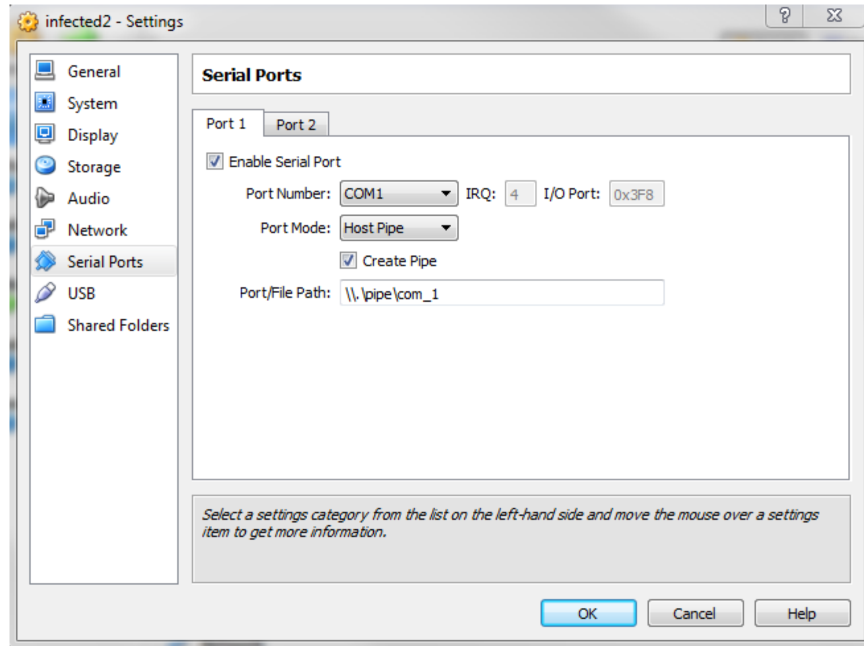
Bu hataların anlaşılabilmesi ve giderilebilmesi için Microsoft kullanıcılara/geliştiricilere bazı yazılımlar sunarlar. Bu tip bir durumun analizi için gerekli işlemler ve yazılımların kurulumu adım adım anlatılacaktır.

NOT: Tüm işlemler sanal makine üzerinden gerçekleştirilecektir. Hatanın oluştuğu makineye erişim için gerekli olan araçlar değişmekle birlikte süreç aynıdır.

Sanal Makine Kurulumu

Tüm işlemler “VirtualBox” üzerine kurduğumuz Windows 7 işletim sistemi üzerinden gerçekleştirilecektir. Bunun için VirtualBox üzerinde gerekli ayarlar yapılmalıdır. Debugger ile sanal makine içerisindeki

Windows 7 işletim sistemine bağlanabilmek için bu ayarlar yapılacaktır. Aşağıdaki resimde görüldüğü gibi bu İşlemeri gerçekleştirebilirsiniz.

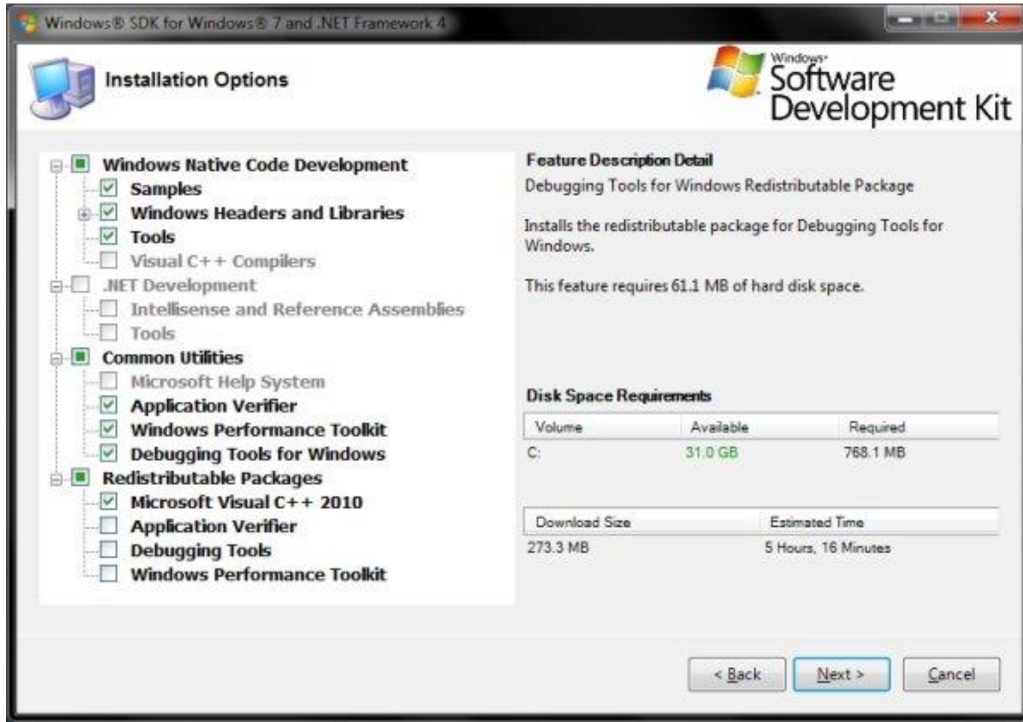


Şekil 21- Windbg için gerekli VirtualBox ayarları.

Yukarıda görülen ayarları tam olarak yaptıktan sonra Windbg ile sisteme bağlanacak port belirtilmiş olacaktır. Windbg için gerekli ayarlamalar bir sonraki adımdan gösterilecektir. Tüm işlemler yapıldıktan sonra sanal makine içerisindeki sisteme "remote" olarak bağlanmak mümkündür.

KD/Windbg Kurulumu

Microsoft geliştiricilerin gerek "user-land" ve gerekse "kernel-land" uygulamaları "debug" edebilmesi için gelişmiş araçlar sunar. Bunlardan bir tanesi, "Debugging Tools for Windows" paketidir, bu paket içerisinde "debugging" için gerekli paketlere ulaşabilirsiniz.

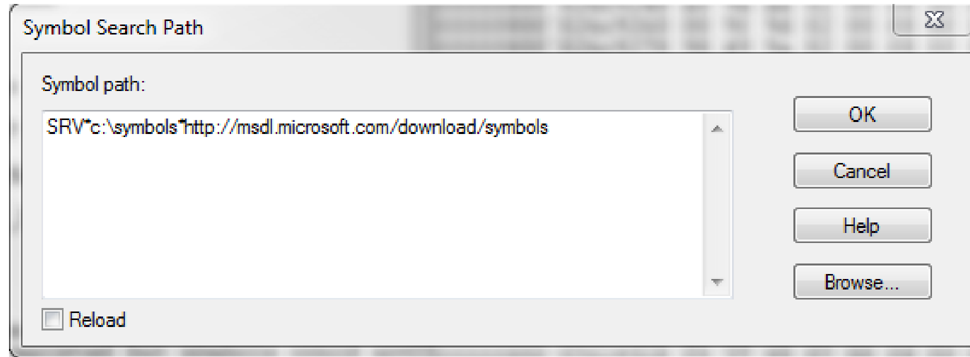


Şekil 22 - Debugging Tools for Windows

Yukarıda görüldüğü üzere "Redistributable Packages" kategorisi altında "Debugging Tools" seçeneği işaretlenmelidir. Böylece gerekli olan araçların kurulumu gerçekleştirilecektir.

KD/Windbg ile Sembol Ayarları

Windbg ile sisteme bağlandıktan sonra Microsoft tarafından sağlanan sembolleri kullanarak, "debugging" işlemini kolaylaştırabilirsiniz. Microsoft tarafından sağlanan bu semboller sayesinde "debug" çıktılarını daha kolay okuyabilir ve böylece daha iyi anlayabiliriz.

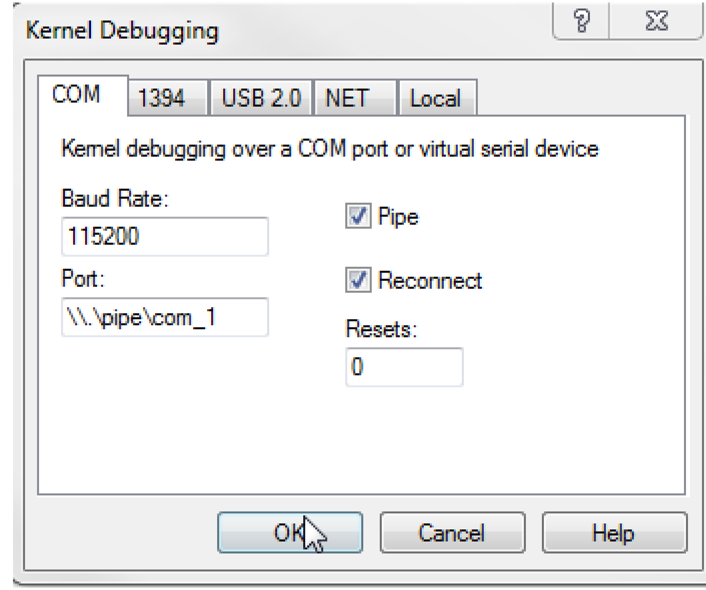


Şekil 23 - Windbg sembol ayaları

Remote Kernel Debugging

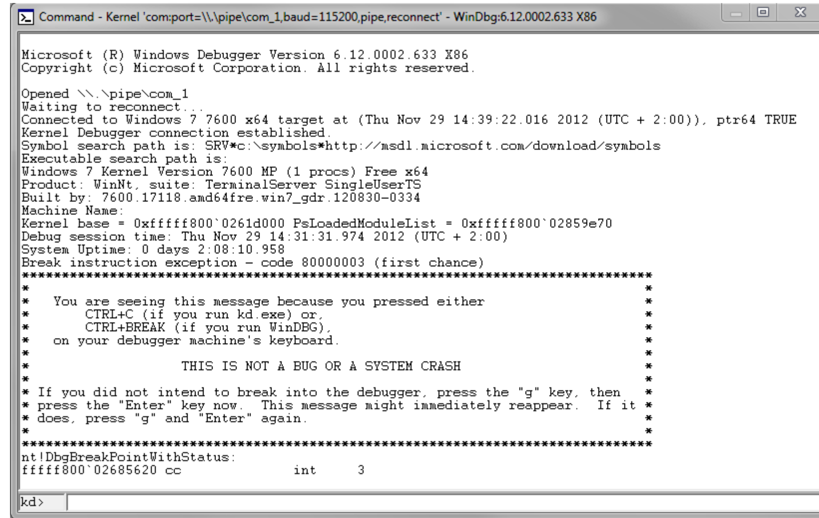
Daha önce VirtualBox içerisinde Windows işletim sistemine bağlanabilmek için bir takım ayarlar yapmıştık. Bu ayarları yapmamızın sebebi Windbg ile uzaktaki bir makineye "kernel mode debugging" yapmak için bağlanabilmektir.

VirtualBox içerisinde belirlediğimizi ayarları Windbg ile bağlanabilmek sırasıyla *“File – Kernel Debugging”* ile açılan pencereye girmemiz gerekiyor.



Şekil 24 - Kernel Debugging penceresi.

Yukarıda görüldüğü gibi sadece “Port” kısmına VirtualBox içerisinde belirlediğimiz bölümü eklememiz gerekiyor. Tüm ayarlarınız yukarıda anlatıldığı gibi ise “OK” butonuna basarak Kernel Debugger’ı aktif hale getirebiliriz.



Şekil 25 - Windbg ile kernel debug.

Bir önceki resimde görüldüğü gibi yukarıda ki ekranın başarılı bir şekilde gelmiş olması lazım. Dikkat edilmesi gereken nokta ilk satırlarda “waiting to reconnect” mesajını görüdüğünüzde “Ctrl + Break” tuşuna basmanız gerekir.

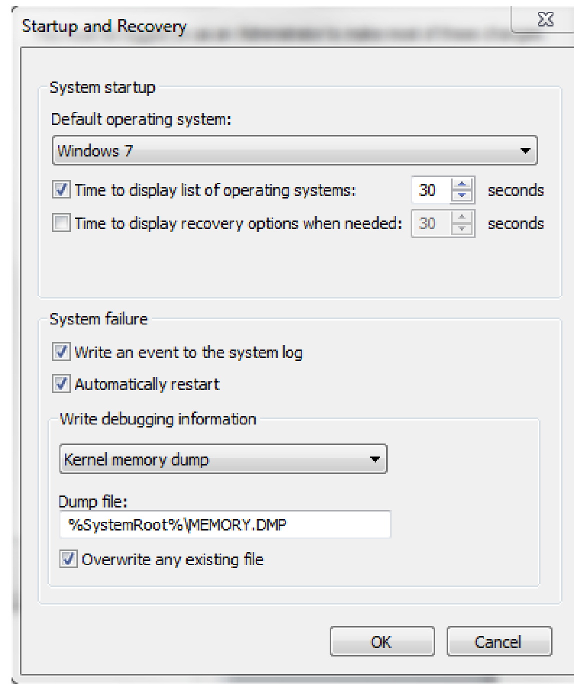
Aynı işlemi komut satırından gerçekleştirebilirsiniz, Debugging Tools for Windows klasörü içerisinde bulunan kd.exe isimli yazılım komut satırı yazılımıdır.

```
kerndebug
Built by: 7600.17118.amd64fre.win7_gdr.120830-0334
Machine Name:
Kernel base = 0xfffff800`0264c000 PsLoadedModuleList = 0xfffff800`02888e70
System Uptime: not available
Break instruction exception - code 80000003 (first chance)
*******************************************************************************
*                                                                             *
*   You are seeing this message because you pressed either                    *
*       CTRL+C (if you run kd.exe) or,                                        *
*       CTRL+BREAK (if you run WinDBG),                                       *
*   on your debugger machine's keyboard.                                      *
*                                                                             *
*                   THIS IS NOT A BUG OR A SYSTEM CRASH                       *
*                                                                             *
* If you did not intend to break into the debugger, press the "g" key, then   *
* press the "Enter" key now.  This message might immediately reappear.  If it *
* does, press "g" and "Enter" again.                                          *
*                                                                             *
*******************************************************************************
nt!DbgBreakPointWithStatus:
fffff800`026b4620 cc              int     3
kd>
```

Şekil 26 - KD.EXE komut satırı aracı.

Windows İşletim Sisteminin Ayarlanması

Windows işletim sistemi üzerinde BSOD analizinin yapılabilmesi için öncesinde bir kaç ayar yapmak gerekir. Bunun için “kontrol paneli” üzerinden “view advanced system settings” seçeneği seçilmelidir. Açılan pencereden “Startup and Recovery” kısmından gerekli ayarlar aşağıda görüldüğü gibi yapılmalıdır.



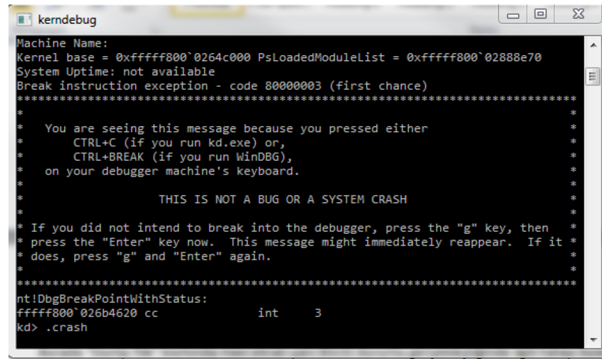
Şekil 27 - BSOD için gerekli ayarlar.

Burada “Dump file” kısmında mavi ekran yani BSOD durumu gerçekleştiğinde ilgili hatayı bulabilmek için ilgili “memory dump” verisinin nereye kaydedileceği bilgisidir. Kernel için “minidump” verisidir, bu dosyaya erişebilmek için “Windows” dizini altında “minidump” isimli klasöre erişmeniz gerekir.

Örnek: Crash Dump Analizi

Örnek bir uygulamanın gerçekleştirilebilmesi için Windows içerisindeki driver kullanılacaktır. Hatanın nereden kaynaklandığını bulmamız gerekmekte, bunun için örnek bir olay yaratmamız gerekiyor. Daha

önce "debugger" ile "kernel" içine girebildiysek vermemiz gereken komut ".crash" komutudur. Böylece örnek bir mavi ekran hatası alacağız.



Şekil 28 - Örnek bir olay analizi.

Bu komutu verdikten sonra sanal makine içerisindeki işletim sisteminin mavi ekran vermesi geremektedir. Bu komut verildikten sonra işletim sistemi tarafında aldığımız mavi ekran hatasının hangi sürücüden kaynaklı olduğunu bulmamız artık kolaydır.

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.


The end-user manually generated the crashdump.

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000E2 (0x0000000000000000,0x0000000000000000,0x0000000000000000,0
x0000000000000000)



Collecting data for crash dump ...
Initializing disk for crash dump ...
Beginning dump of physical memory.
Dumping physical memory to disk:  100
Physical memory dump complete.
Contact your system admin or technical support group for further assistance.
```

Şekil 29 - Örnek bir mavi ekran hatası.

Yukarıda resimde görülen "Dumping physical memory to disk" yazısı eğer siz aynı işlemleri denediğinizde çıkmadı ise "g" komutu ile bu işlemi başlatabilirsiniz. Bunun anlamı daha önce işletim sistemi üzerinde ayarladığımız "memory dump" ile ilgilidir. Memory dump verisini aldıktan sonra tüm işlemleri kolayca sürdürebiliriz.